ASM Sc. J., 11, Special Issue 3, 2018 for SANREM, 172-177

Minimizing Computational Cost in Mesh Subdivision by using Doubly-Connected Edge List with Modified Butterfly Schemes

Nurhamiezrah Midin[1*], Abdullah Bade[1] and Hamzah Asyrani Sulaiman[2]

[1]*Mathematics, Graphics and Visualization Research Group (M-GRAVS), Faculty of Science and Natural Resources, Universiti Malaysia Sabah.*

[2]*Media Interactive Department, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia.*

Surface subdivision is crucial in graphic application especially in mesh modelling to produce smooth surface and a realistic visualization of a 3D object. Triangular mesh is a simplest mesh that used widely in 3D object modelling. Modified Butterfly Scheme is one of the mesh subdivision scheme which adds vertices and modifies mesh topological information to produce finer mesh from coarse mesh. Complexity increases as the number of vertices increases thus leads to expensive computational cost. Doubly-Connected Edge List is a data structure used to store mesh data in a convenient way so that the mesh information can be easily accessed. This paper aims to use Doubly-Connected Edge List in Modified Butterfly Scheme to reduce the execution time needed to compute the subdivision process of triangular meshes. It does not include calculation of mesh volume as the process only applied on the surface only. Experiments are conducted on different 3D models with various numbers of triangular faces. The result shows that by using both Doubly-Connected Edge List and Modified Butterfly Scheme, time taken to compute 3D bunny (1168 faces) is 189.9 seconds (4.385 seconds faster) compared to the Modified Butterfly Scheme alone. Although the time taken for suggested technique takes 0.44932 seconds slower than the Modified Butterfly Scheme in computing the tetrahedron (4 faces), as the number of faces increases, it shows that the suggested technique performs better in models computation. It is believed that this integration will help in minimizing the computational cost for surface subdivision.

Keywords: Modified Butterfly Scheme, Doubly-Connected Edge List, triangular mesh, surface subdivision, computational time.

I. INTRODUCTION

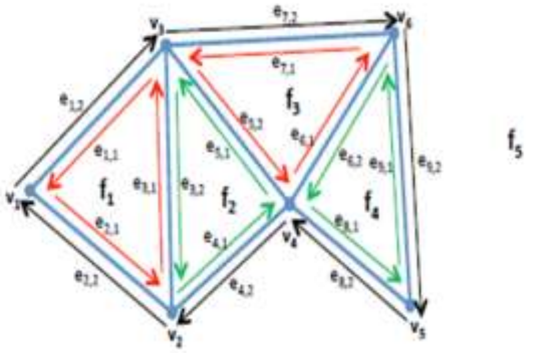
In computer graphics, objects are often represented by triangle mesh (KH 2002). With the advances in data acquisition and the development of modeling techniques, 3D models become more and more complex. Although the rendering capability of current graphic hardware has been improved considerably, it can not yet keep up with the growing of model size such in subdivision process, which makes real-time rendering difficult. Subdivision process is widely used to produce smoother and more detailed 3D models (Pakdel & Samavati, 2007).

*Corresponding author's e-mail: miezramidin@gmail.com

Generally, in subdivision surfaces, the subdivision process is over a simple polyhedron. The whole polyhedral meshes are refined globally at a level of mesh density. After several subdivision steps, the generated subdivision surface will be smooth enough to represent a fine shape (Jia *et al.*, 2012). But even after a few subdivision steps, the number of generated meshes will become huge. In many applications, the highly detailed polygonal models are necessary and the rendering time is important.

II. MATERIALS AND METHODS

Doubly-Connected Edge List (RH 2013) stores 3 types of mesh data, vertices, half-edges and faces data. Each of the vertex hold its half-edge information. It is called as incident pointer. Every half-edge associated with only one vertex and one face. They have next pointer which point the next half-edge and previous pointer that point them. Each triangle basically has 3 pairs of half-edges. Twins of half-edges will form an edge. The sequences of half-edge that linked by their pointer will form a face. Figure 1 illustrates the data representation in the Doubly-Connected Edge List (DCEL).



Half-edge	Origin	Twin	IncidentFace	Next	Previous
$e_{3,1}$	v_2	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,1}$
$e_{3,2}$	v_3	$e_{3,1}$	f_2	$e_{4,1}$	$e_{5,1}$
$e_{4,1}$	v_2	$e_{4,2}$	f_2	$e_{5,1}$	$e_{3,2}$
$e_{4,2}$	v_4	$e_{4,1}$	f_5	$e_{2,2}$	$e_{8,2}$
–	–	–	–	–	–

Figure 1. Data representation in DCEL

The process of storing the read data starts with initializing the vertices by setting their ID. The vertices then linked by edges to form separate triangles. Triangles are connected later by creating edges to link all the triangles until closed faces are formed. The original algorithm of DCEL has edge iterator which iterate over the half-edges of the contour of that face if it receives a face pointer in the constructor. If it receives a vertex pointer, it will iterate over the half-edges that starts on that vertex. There is also method to load and save DCEL structures into files, which is able to deal with the DCEL structure and the data stored in it. However, these steps are not included in this paper to reduce the computational costs.

Butterfly scheme is an interpolating scheme for triangular mesh which interpolates its control points. It means that the control points are remained on the limit surface after few iteration of refinement took place. It then modified to produce C^1 continuity to produce smoother triangular limit surfaces. After a rough wireframe of a 3D object is modelled by using the data recorded by the DCEL, mesh subdivision is applied on the surface of the wireframe to make it smoother by increasing the number of polygons.

This paper combines Butterfly Scheme (Dyn *et al.*, 1990) and Modified Butterfly Scheme (Zorin *et al.*, 1996) with DCEL. Butterfly Scheme only handles the ordinary vertex with valence 6 and Modified Butterfly Scheme manages the extraordinary vertex with valence less than 6. By combining both schemes, smoother surface of an object is obtained. The schemes add vertices at the midpoint of the half-edges for every triangle. There are three consideration before adding the vertices. Each case requires different rules.

I. Half-edge connects two endpoints of valence 6
II. One of the endpoint of the half-edge is extraordinary point with K-vertex (K≠6)
III. Both of the endpoints are extraordinary points

During the refinement process, a vertex is added along each half-edges of the triangles using the stencils as shown in Figure 2 and Figure 3. The stencils used for ordinary and extraordinary vertex (vertex valence = 3, 4, 5, 6). Valence of the vertex determines the stencil type and rule. Then, each face is split into four subtriangles using the new vertices. This process is repeated recursively until a desire smooth surface is obtained.

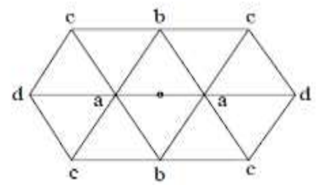


Figure 2. Stencil for ordinary valence (6)

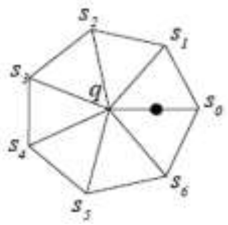


Figure 3. Stencil for extraordinary valence (3, 4, 5)

For edges connect two regular vertices, the canonical setting and weight are given by a = $\frac{1}{2} - w$, b = $\frac{1}{8} + 2w$, c = $-\frac{1}{16} - w$, d = w where $w = 0$. For edges connecting a K-vertex (K≠6)

I. K = 3, $s_0 = 5/12$, $s_{1,2} = -1/12$
II. K = 4, $s_0 = 3/8$, $s_2 = -1/8$, $s_{1,3} = 0$
III. K ≥ 5, $s_j = \left(\frac{\frac{1}{4} + (\cos\frac{2\pi j}{K}) + \frac{1}{2}\cos(\frac{4\pi j}{K})}{K}\right)$

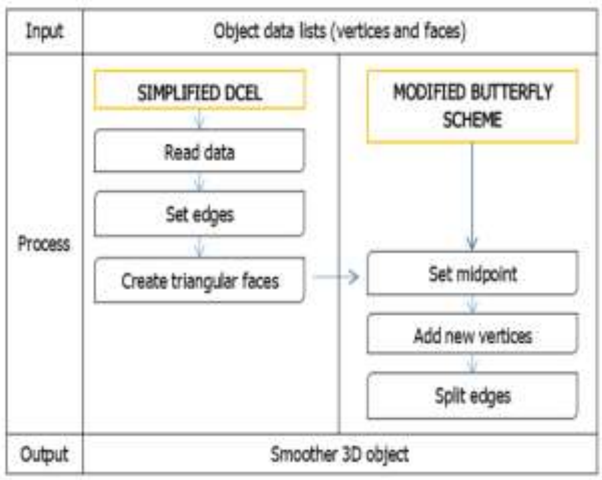


Figure 4. System flow

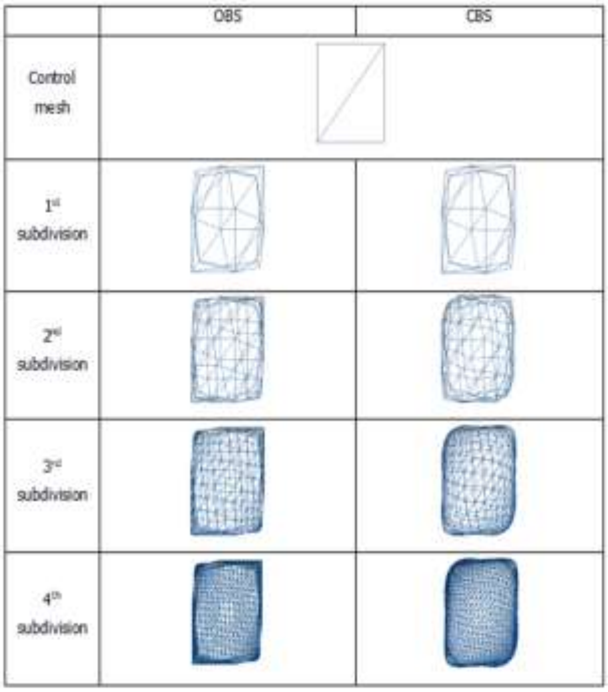
Figure 4 illustrates the system's flow from input, process and the output. Based on the figure, the input file of an object only consists of vertices and faces data without its other data such as vertex normal and texture. After the object file loaded into the system, DCEL will read and compute the data to connect each of the vertices by creating half-edges and triangular faces to produce surface of a 3D object. Subdivision process then occurs starting with the setting of midpoints on the half-edges before new vertices are added on the surface. New position of the vertices will make the half-edges splitted into two half-edges for each of the half-edge which produce smoother 3D model. Smoother and better visualization of a 3D model will be the output.

III. RESULT AND DISCUSSION

Experiments are conducted in Microsoft Visual Studio 2010 by using C++ language. 6 different 3D triangular mesh with various number of vertices are used in the experiments. First experiment is to compare the visual appearance for objects using original Modified Butterfly Scheme (OBS) with objects that use both combined Butterfly Scheme and Modified Butterfly Scheme (CBS). Second experiment compares the execution time for CBS and CBS after DCEL integrated in it.

Table 1 shows the experiment done on a cube with 8 vertices. It shows that by combining the Butterfly Scheme and Modified Butterfly Scheme, smoother surface of an object can be obtained after few iteration of subdivision. Comparing the two methods, OBS and CBS, it is clearly shows that edge on the objects' corner are curve if the combines scheme is applied compared to the original one. The curves that been produced by the combined schemes make the object looks smoother.

Table 1. Curve visualization



As mentioned before, Modified Butterfly Scheme only managed to handle vertex with valence 3, 4 and 5. By combining the Modified Butterfly Scheme with the Butterfly Scheme, it seems better to use both schemes so that more valence vertex could be handled: the extraordinary vertices (valence 3, 4 and 5) and the ordinary vertices (valence 6). After CBS is developed, DCEL data structure is applied in CBS to reduce the computational time. Table 2 records the execution time for CBS and the integrated DCEL in CBS.

Table 2. Comparison of average computational time for 3D model

MODEL	AVERAGE COMPUTATIONAL TIME (s)	
	CBS	CBS + DCEL
Tetrahedron (4 faces)	1.37185	1.82117
Cube (12 faces)	1.39134	1.83145
Torus (200 faces)	8.93901	8.39953
Sphere (224 faces)	10.86732	10.01551
Dress (380 faces)	29.60934	26.73834
Bunny (1168 faces)	194.285	189.900

The result indicates that as the number of vertices getting higher, the integrated scheme performs faster than the CBS alone. It is believe that CBS will be more efficient if DCEL is used as the data structure in the scheme. Figure 5 shows the objects used in the experiments. All of the objects are 3D objects and constructed by triangular mesh.

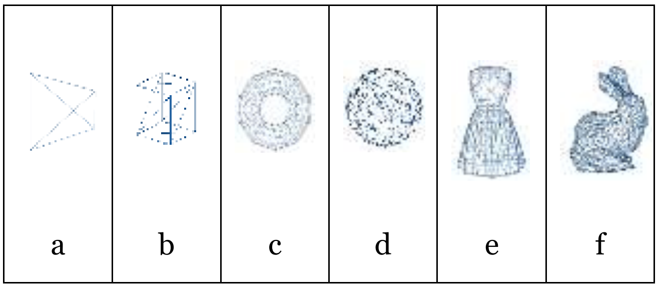


Figure 5. a) tetrahedron, b) cube, c) torus, d) sphere, e) dress, f) bunny

IV. CONCLUSION

As a conclusion, the aim to reduce the execution time needed to compute the subdivision process of triangular meshes by using DCEL in CBS is accomplished. The Modified Butterfly Scheme is first combined with the traditional Butterfly Scheme to get smoother surface of an object before it is integrated with Doubly-Connected Edge List.

V. ACKNOWLEDGEMENTS

The authors would like to express sincere gratitude to the Media Interactive Department, FICT-UTeM for funding this research and to M-GRAVS for providing supervisory support.

[1] Dyn, N., Levin, D. & Gregory, J. A. (1990). A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. ACM Transactions on Graphics, 9, 160-169.

[2] Jia, Y., Wang, X., Ma, X., Li, J. & Zhou, X. (2012). Local Subdivision on Triangle Mesh. International Conference on Audio, Language and Image Processing (ICALIP), 356-359.

[3] Kai, H. (2002). An Easy Way of Detecting Subdivision Connectivity in a Triangle Mesh, Technical Report no.3, Department

of Computer Science 9, University of Erlangen.

[4] Muller, D. E. & Preparata, F. P. (1978). Finding the Intersection of Two Convex Polyhedra. Theoretical Computer Science, 7, 217-236.

[5] Pakdel, H. R. & Samavati, F. F. (2007). Incremental Subdivision for Triangle Meshes. International Journal of Computational Science and Engineering. 3 (1), 80-92.

[6] Ryan, H. (2013). The DCEL Data Structure for 3D Graphics, viewed 15 July 2017, <http://www.holmes3d.net/graphics/dcel />

[7] Zorin, D., Schr¨oder, P. & Sweldens, W .(1996). Interpolating Subdivision for Meshes of Arbitrary Topology. Tech. Rep. CS-TR-96-06, Caltech, Department of Computer Science, Caltech.